

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

This Page Blank (uspto)

(19) RÉPUBLIQUE FRANÇAISE
INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE
PARIS

(11) N° de publication :
(à utiliser que pour les commandes de reproduction)

2 612 316

(21) N° d'enr. gisrement national :

88 03206

(51) Int Cl⁴ : G 06 K 19/06.

(12)

DEMANDE DE BREVET D'INVENTION

A1

(22) Date de dépôt : 11 mars 1988.

(30) Priorité : JP, 13 mars 1987, n° 59808/1987.

(71) Demandeur(s) : Société dite : MITSUBISHI DENKI KA-BUSHIKI KAISHA. — JP.

(72) Inventeur(s) : Kenichi Takahira.

(43) Date de la mise à disposition du public de la demande : BOPI « Brevets » n° 37 du 16 septembre 1988.

(60) Références à d'autres documents nationaux apparentés :

(73) Titulaire(s) :

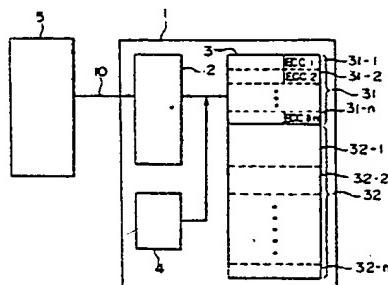
(74) Mandataire(s) : Cabinet Weinstein.

(54) Carte à circuits intégrés ayant une capacité de vérification d'erreur interne.

(57) L'invention concerne une carte à circuits intégrés particulièrement adaptée à des fonctions multiples et ayant une capacité de vérification d'erreur.

Selon l'invention, la carte 1 possède une mémoire 3, 4 et un microprocesseur 2, la mémoire contient un certain nombre de blocs d'applications 32 auxquels l'accès est sélectivement permis par le microprocesseur et un bloc protégé qui est accessible par le microprocesseur mais inaccessible par un terminal 5; dans le bloc protégé sont stockées des données spécifiant l'emplacement, la taille et l'application associées à chaque bloc d'applications ainsi qu'un code de vérification d'erreur se rapportant aux données stockées dans le bloc d'applications.

L'invention s'applique notamment aux cartes à circuits intégrés permettant d'accomplir des transactions complexes.



FR 2 612 316 - A1

La présente invention se rapporte à une carte à circuits intégrés du type possédant un microprocesseur et une mémoire, et elle se rapporte plus particulièrement à une carte à circuits intégrés multifonctions.

5 Dans de telles cartes à circuits intégrés, le microprocesseur qu'elles possèdent est adapté à contrôler l'accès, par un terminal externe, à la mémoire. Plus typiquement, l'accès se fait par des zones de lecture de la mémoire spécifiques au microprocesseur et l'émission 10 de l'information souhaitée par ses lignes d'entrée/sortie pour couplage au terminal. Cependant, des transferts plus directs de l'information entre la mémoire et un dispositif externe sous le contrôle du microprocesseur sont également possibles.

15 Des cartes à circuits intégrés sont développées pour accomplir des transactions de plus en plus complexes, ou groupes de transactions, et en faisant ainsi, utilisent de plus en plus la puissance de traitement et la capacité de stockage rendues disponibles 20 par la micro-électronique moderne. Bien que les capacités des cartes à circuits intégrés soient ainsi croissantes, avec la possibilité d'accomplir des types multiples de diverses transactions en connexion avec le même type ou 25 des types multiples de divers terminaux, les demandes imposées sur les capacités de la carte à circuits intégrés deviennent de plus en plus sévères. En général, dans une carte à circuits intégrés d'applications multiples, la carte est conçue pour répartir l'espace de mémoire qu'elle possède en blocs séparés d'applications 30 et pour permettre à une application particulière (par exemple, d'accomplir un type de transaction avec un type particulier de terminal) de n'accéder qu'au bloc ou aux blocs d'applications dans la mémoire, assignés à ce type de transaction. Comme les types des transactions sont 35 différents, par leur étendue, leur complexité et la

quantité de la mémoire requise, il est souhaitable de permettre au microprocesseur de séparer la mémoire en blocs de diverses tailles.

Dans un système d'une carte à circuits intégrés, des séquences d'opération sont usuellement accomplies entre l'unité de la carte à circuits intégrés et la personne qui possède la carte ainsi qu'entre l'unité de la carte et un terminal qui commande directement la carte. Si la carte est insérée dans le terminal par l'opérateur, le terminal applique une horloge de source de courant ou analogue afin d'actionner la carte, permettant ainsi la discrimination, l'interclassement, l'identification du possesseur de la carte, et ainsi de suite, entre la carte à circuits intégrés et le terminal.

Quand ces opérations préliminaires ont été accomplies, le terminal identifie un bloc spécifique de la mémoire de la carte à circuits intégrés et obtient l'accès à ce bloc afin d'accomplir l'application souhaitée. Tandis que diverses méthodes sont disponibles pour l'accès au bloc et pour accomplir l'application, le procédé comprend, dans chaque cas, la lecture du bloc accédé de la mémoire.

En général, dans le cas d'une carte conventionnelle à circuits intégrés à plusieurs applications, la gestion générale des erreurs d'enregistrement de données (vérification d'erreur de donnée) est accomplie par le terminal seul. En effet, la carte à circuits intégrés a, dans des blocs d'applications de sa mémoire, des articles d'information de répertoire pour accéder aux articles des données enregistrées dans la mémoire et des codes de vérification d'erreur pour la gestion des erreurs d'enregistrement de données. Tandis que la carte peut lire ou écrire les données d'enregistrement en utilisant le répertoire, elle

ne peut reconnaître ou interpréter un code de vérification d'erreur enregistré en tant que la fin des articles de données enregistrées dans chaque bloc. Pour effectuer la gestion des erreurs de données
5 d'enregistrement, un moyen pour rechercher un code de vérification d'erreur, contenu dans les articles des données enregistrées, un moyen pour accomplir la vérification d'erreur sur la base du contenu des données enregistrées et du code de vérification d'erreur et un moyen pour produire un code de vérification d'erreur à partir des articles de données à enregistrer sont nécessaires. Ainsi, dans le cas d'une carte conventionnelle à circuits intégrés, le terminal contient ces moyens et accomplit la gestion des erreurs de données
10 d'enregistrement par accès à la carte, selon la procédure de traitement programmée dans le terminal. Dans ce cas, pour changer partiellement un bloc de données, il est nécessaire que le terminal extrait la totalité du bloc correspondant de données enregistré dans la carte à circuits intégrés (données d'enregistrement et code de vérification d'erreur) afin de produire un code de vérification d'erreur pour ce bloc de données car un code de vérification d'erreur est établi dans chaque bloc de données de la carte à circuits intégrés établi par le
15 terminal. Si ce terminal appartient à un groupe d'un certain nombre de systèmes d'applications, il doit être pourvu de groupes de moyens des types ci-dessus mentionnés adaptés à différents types de codes de vérification d'erreur ou méthodes d'établissement de code
20 d'erreur spécifique pour ces systèmes d'applications. Par conséquent, si le terminal accomplit la gestion des erreurs de données d'enregistrement, cela augmente la charge imposée au terminal. De plus, il y a une possibilité qu'une donnée enregistrée et un code correspondant de vérification d'erreur dans la carte à
25
30
35

circuits intégrés conventionnelle soient par exemple changés intentionnellement car, comme on l'a décrit ci-dessus, la donnée enregistrée et le code de vérification d'erreur sont tous deux extraits par le 5 terminal afin de traiter la donnée enregistrée. Ce changement ne peut être détecté lors d'une vérification ultérieure. Il y a par conséquent un problème en termes de sécurité de données. Un exemple de la carte à circuits intégrés conventionnelle ayant une fonction de 10 vérification de sa propre donnée enregistrée est connu. Dans ce cas, il est nécessaire d'attacher un code de vérification d'erreur à chaque article de données. La proportion des zones de mémoire occupées par les codes de vérification d'erreur est ainsi accrue et cela est 15 désavantageux en termes d'efficacité d'utilisation de la mémoire de la carte à circuits intégrés.

Etant donné ce qui précède, la présente invention a pour but général une carte à circuits intégrés pouvant s'adapter à diverses applications, mais 20 accomplissant la gestion des erreurs de données sans charge significative ajoutée à l'équipement terminal.

De ce point de vue, la présente invention a pour objet d'accomplir une vérification d'erreur sur la carte à circuits intégrés elle-même, en utilisant le 25 processeur qu'elle possède, tout en permettant encore une grande flexibilité de répartition de la mémoire en blocs de diverses dimensions.

La présente invention a pour objectif plus détaillé une carte à circuits intégrés où la sécurité des 30 données stockées est améliorée tout en permettant encore un plein accès au terminal pour la donnée réelle d'applications.

A cette fin, on prévoit une carte à circuits intégrés adaptée à une interface avec un terminal pour 35 accomplir un certain nombre d'applications. La carte

possède une mémoire divisée en un certain nombre de blocs d'applications de taille variable (ou différentes) pour le stockage des données d'applications et un autre bloc qui est un bloc protégé, c'est-à-dire auquel le terminal ne peut accéder. Le bloc protégé a un emplacement se rapportant à chacun des blocs d'applications et chaque emplacement est adapté à stocker une donnée d'identification et un code de vérification d'erreur pour le bloc associé d'applications. Le microprocesseur commande l'accès à la mémoire de manière à prévenir l'accès au bloc protégé par le terminal et à permettre sélectivement l'accès aux blocs d'applications par le terminal. Avant de permettre l'accès du terminal à un bloc d'applications, la vérification d'erreur de la donnée stockée pour l'application choisie est accomplie. Lorsqu'une nouvelle donnée est introduite dans la mémoire de la carte à circuits intégrés dans le bloc d'applications, le code calculé de vérification d'erreur pour la nouvelle donnée est stocké dans le bloc protégé à l'emplacement associé au bloc choisi d'applications.

Selon la présente invention, dans la carte à circuits intégrés, est incorporé un groupe de moyens pour accomplir la vérification d'erreur des données d'enregistrement, comprenant un moyen pour produire un code de vérification d'erreur et un moyen pour accomplir la vérification d'erreur en utilisant le code de vérification d'erreur et les données. Si la carte à circuits intégrés est conçue pour accomplir également la correction de données, elle contient de plus un moyen de correction de données. Lorsque la donnée est enregistrée dans un bloc prédéterminé d'applications de la mémoire dans la carte à circuits intégrés, le moyen produisant le code de vérification d'erreur produit simultanément, de cette donnée d'enregistrement, un code de vérification d'erreur s'y rapportant et ce code de vérification

d'erreur est enregistré dans un bloc protégé correspondant au bloc d'applications où la donnée est enregistrée. La vérification d'erreur de la donnée d'enregistrement est accomplie par le moyen de 5 vérification d'erreur sur la base de la donnée enregistrée dans la bloc d'applications et du code de vérification d'erreur correspondant à ce bloc. Pour effectuer également une correction de donnée, le moyen de correction décode les contenus du résultat de la 10 vérification accomplie par le moyen de vérification d'erreur, et corrige les erreurs dans la donnée. Si la donnée est réécrite, un nouveau code de vérification d'erreur pour la donnée réécrite dans le bloc 15 d'applications est produit et est enregistré dans le bloc protégé correspondant. Dans la pratique, un programme pour entreprendre le traitement est stocké dans une mémoire morte prévue dans la carte à circuits intégrés et est exécuté par le microprocesseur.

L'invention sera mieux comprise et d'autres 20 buts, caractéristiques, détails et avantages de celle-ci apparaîtront plus clairement au cours de la description explicative qui va suivre faite en référence aux dessins schématiques annexés donnés uniquement à titre d'exemple 25 illustrant un mode de réalisation de l'invention et dans lesquels :

- la figure 1 donne un schéma bloc montrant une carte à circuits intégrés construite selon la présente invention, en interface avec une unité externe au terminal ; et

30 - la figure 2 donne un schéma bloc d'une séquence d'opérations pour l'utilisation d'une carte à circuits intégrés selon la présente invention.

L'invention sera maintenant décrite en se référant à un mode de réalisation préféré.

En se référant maintenant aux dessins, la figure 1 montre une carte à circuits intégrés 1 selon la présente invention, connectée par un chemin de données 10 à un terminal 5. Le terminal 5 contient typiquement un lecteur/scripteur, a son propre processeur interne et a également fréquemment une liaison de communications avec un ordinateur central ou une base de données.

La carte à circuits intégrés 1 possède son propre microprocesseur 2 et son moyen formant mémoire que l'on peut voir sur les dessins sous la forme d'une mémoire de données 3 et une mémoire de programme ou mémoire morte 4 qui sont séparées. La mémoire de données 3 peut être une mémoire à accès aléatoire, une mémoire morte programmable électriquement, une mémoire morte programmable et effaçable électriquement, tandis que la mémoire de programme peut être une mémoire morte ou une mémoire morte programmable électriquement ou une mémoire morte programmable et effaçable électriquement. De manière pratique, les mémoires 3, 4 sont combinées dans le même dispositif à mémoire et, encore mieux, elles sont toutes deux incorporées sur la même pastille que le microprocesseur de manière que la carte à circuits intégrés 1 ne nécessite qu'un seul semi-conducteur à y noyer sans avoir besoin de connexions entre semi-conducteurs.

Avant de mieux décrire la structure à l'intérieur de la carte à circuits intégrés ainsi que le moyen pour séparer la mémoire, il faut d'abord noter que le chemin de données 10 forme une connexion pour le transfert de données entre le terminal 5 et la carte à circuits intégrés 1. Un exemple d'une telle connexion est la carte à circuits intégrés et le terminal sans contact dont on dispose aujourd'hui, où des bobines d'accouplement sur la carte à circuits intégrés et le terminal sont mises en juxtaposition lorsque la carte est

insérée dans le terminal de manière que les bits de données puissent être transférés de la carte au terminal ou du terminal à la carte par couplage magnétique entre les bobines juxtaposées.

5 Le terminal ne sera pas décrit en détail car il est conventionnel et bien connu, mais contient typiquement son propre processeur pour accomplir des opérations locales et une liaison de communications vers une base centrale de données qui maintient un dossier central pour des applications à accomplir par tous les utilisateurs qui ont accès au système.

10 En se référant de nouveau à la carte à circuits intégrés 1, la mémoire morte 4 contient un programme stocké qui commande le microprocesseur 2 pour forcer la carte à circuits intégrés 1 à accomplir les fonctions logiques qui lui sont assignées. Typiquement, la mémoire morte 4 est une section comparativement limitée de mémoire qui nécessite une programmation précise et simple afin de rendre maximale l'efficacité de la carte 1.

15 20 La mémoire de données 3 est typiquement bien plus grande que la mémoire morte 4 mais doit également être efficacement utilisée afin d'obtenir l'efficacité maximale de la carte 1. La plus grande partie de la mémoire de données 3 est vouée à une zone d'applications 25 32 qui est divisée en une série de blocs d'applications 32-1, 32-2, ... 32-n. La mémoire de données 3 est également séparée ou divisée pour produire un assez petit bloc protégé 31 qui contient l'information de répertoire pour chacun des blocs d'applications ainsi qu'une 30 information de code de vérification d'erreur pour chacun des blocs d'applications. La zone 31 est protégée dans le sens que le microprocesseur 2 est programmé pour empêcher l'accès à la zone protégée 31 par le terminal 5. Le microprocesseur 2, cependant, a accès aux données dans la 35 zone protégée 31 mais n'utilise cette information qu'à

l'intérieur de la carte à circuits intégrés. La zone 31 est protégée par programmation de la mémoire morte 4 de manière à empêcher le microprocesseur 2 de s'adresser aux emplacements dans le bloc 31 en connexion avec les étapes 5 de programme qui permettent le transfert de données entre le terminal 5 et la mémoire de données 3.

La figure 1 montre que la zone d'applications 32 est divisée en un certain nombre de blocs d'applications 32-1, 32-2, ...32-n qui ont usuellement des tailles différentes. Dans certaines circonstances, 10 les blocs peuvent être prérépartis ou divisés avant que la carte ne soit mise en utilisation ou bien le microprocesseur 2, en conjonction avec son programme interne et l'information reçue du terminal 5, peut 15 diviser les blocs pendant le cours de l'accomplissement des applications.

Dans la zone protégée 31, et en correspondance avec chacun des blocs d'applications se trouve un emplacement identificateur, comme des emplacements de mémoire 31-1, 32-2, ...32-n. Dans le cas de la zone 20 protégée 31, chacun des emplacements peut être de la même taille et il y a un emplacement pour chacun des blocs d'applications. Comme le montre la figure 1, chacun des blocs d'identification ou mots contient une information 25 de répertoire et un code de vérification d'erreur pour le bloc associé d'applications. Par exemple, l'information de répertoire peut contenir un groupe de bits spécifiant un numéro d'identification de l'application particulière, un autre groupe de bits spécifiant l'adresse de départ du 30 fichier d'applications assigné à cette application et un autre groupe de bits spécifiant la taille de ce fichier d'applications. Ainsi, lorsque la carte à circuits intégrés 1 est insérée dans un terminal 5 et que le terminal demande l'accès à une application particulière, 35 le microprocesseur 2 peut rechercher la zone protégée 31

pour le numéro d'identification de l'application requise puis a immédiatement accès à l'emplacement de départ et la taille du fichier dans le champ d'applications 32 assigné à cette application particulière.

5 De plus, et selon l'invention, le processeur a également accès à un code de vérification d'erreur stocké dans une zone protégée se rapportant à la donnée particulière qui est alors stockée dans le fichier associé d'applications.

10 Selon la présente invention, dans la ROM 4 est également stocké un programme de production d'un code de vérification d'erreur et un programme pour accomplir la vérification de l'erreur. Le microprocesseur 2 entreprend la formation du code de vérification d'erreur et la vérification des erreurs de données. Si la donnée est enregistrée dans un bloc d'applications (comme le bloc d'applications 32-1 dans la zone d'applications 32), un code de vérification d'erreur se rapportant à cette donnée est produit par le microprocesseur 2 et ce code de vérification d'erreur est enregistré dans l'emplacement de mémoire 31-1 de la zone protégée 31 correspondant au bloc d'applications 32-1. Au moment de la vérification de l'erreur, le microprocesseur 2 vérifie la donnée enregistrée dans le bloc d'applications sur la base du code correspondant de vérification d'erreur. Diverses méthodes sont applicables à la vérification de l'erreur. Un exemple d'une méthode bien connue se rapportant à une vérification par redondance cyclique est décrite dans "Date Commnunication Handbook" par Denshi Tsushin Gakkai (Octobre 1984, pages 49-53). Dans cette méthode, un déterminant est formé d'articles de données en série sur la base d'une application d'un principe qui réside dans le fait qu'une matrice unitaire est obtenue en multipliant un déterminant par sa matrice inverse. Si le résultat de cette opération (usuellement appelée un

15

20

25

30

35

syndrome) est zéro, il n'y a pas d'erreur de données. S'il n'est pas de zéro, l'existence des erreurs de données peut être détectée. Pour effectuer également la correction des données, il est nécessaire de stocker au préalable un programme de correction de données dans la mémoire morte 4. Dans ce cas, l'état de l'erreur de données peut être analysé à partir du syndrome. Les erreurs de données sont corrigées en décodant le contenu du syndrome. Lorsque la donnée est ré-écrite, un nouveau code de vérification d'erreur pour la donnée dans le bloc d'applications est produit et est enregistré à l'emplacement prédéterminé dans la zone protégée correspondante (c'est-à-dire que le code de vérification d'erreur est ré-écrit).

Le fonctionnement de la carte à circuits intégrés 1 selon la présente invention sera décrit ci-dessous en se référant à l'organigramme montré à la figure 2. On notera que le processus illustré à la figure 2 ne représente qu'une partie d'une transaction d'une carte à circuits intégrés et n'illustre pas, par exemple, les étapes conventionnelles d'insertion de la carte dans le terminal et d'accomplissement des vérifications préliminaires nécessaires d'identification. Comme le montre la figure 2, après avoir accompli l'authentification préliminaire, une étape 20 assigne un fichier d'applications dans les blocs d'applications 32 pour un accès par le terminal. Le type d'applications à accéder est identifié et, au cours de l'accomplissement de l'étape 20, l'information de répertoire dans la zone 31 est recherchée pour trouver l'emplacement ayant un numéro d'identification d'applications correspondant à celui à assigner. Ainsi, le microprocesseur 2 a, à ce moment, accès à l'adresse du bloc d'applications et à la taille du bloc.

Dans la mise en pratique de l'invention, à l'étape 21, le microprocesseur a également accès à et lit le code de vérification d'erreur (ECC) qui a été au préalable introduit dans l'emplacement associé à ce 5 fichier en se basant sur la donnée alors stockée dans le fichier. Une étape 22 est alors accomplie sur la donnée résidant dans le fichier d'applications, en détectant une erreur sur la donnée dans le fichier. Il faut remarquer que la taille du fichier d'applications peut être 10 différente de celle des autres fichiers mais elle est toujours commandée par le même processeur et avec le même programme de détection d'erreur. Si une erreur est détectée à l'étape 23, le programme passe à une étape 28 pour le traitement de l'erreur ou la correction de 15 l'erreur. Sous sa forme la plus simple, le traitement de l'erreur peut prendre la forme d'un simple avortement de la transaction et retour de la carte à l'utilisateur, et ultérieurement, la carte est retirée par une sortie.

Si aucune erreur n'est détectée, le micro- 20 processeur 2 rend alors le fichier disponible au terminal pour la lecture et l'écriture de l'information en 24. Le terminal peut lire une certaine information du fichier auquel il a accès pour déterminer des paramètres de départ pour la transaction à accomplir et à la fin de la 25 transaction, peut écrire l'information remise au point dans les mêmes champs ou des champs différents du fichier auquel il a eu accès. A la suite de l'interaction entre le terminal et le bloc de mémoire auquel il a accès, ce qui est commandé par le microprocesseur 2, une étape 25 est accomplie pour déterminer si une nouvelle donnée a été introduite dans la mémoire sur la carte. Si aucune donnée n'a été introduite, une étape 26 signifie que l'accès au fichier est terminé et si la transaction est alors terminée, la carte est retournée à l'utilisateur. 30 Cependant, si la donnée a été écrite, une étape 27 est 35

accomplie, uniquement dans la carte à circuits intégrés, par laquelle le microprocesseur 2 accomplit un calcul sur la totalité du champ de données dans le bloc choisi d'applications. Ce code de vérification d'erreur, en 5 accomplissant l'étape 27, est alors introduit dans la zone protégée 31 de la mémoire à l'emplacement réservé au code de vérification d'erreur pour le bloc d'applications en question. Ainsi, le nouveau code est disponible dès que le champ d'applications est choisi pour assurer 10 l'intégrité continue de la donnée. Après avoir calculé et enregistré le code de vérification d'erreur par l'étape 27, l'étape 26 signale de nouveau que l'accès au fichier est terminé et la commande est ramenée soit au programme 15 d'applications pour accéder à différents fichiers ou bien la carte est retournée à l'utilisateur.

Dans le mode de réalisation ci-dessus décrit, la lecture du code de vérification d'erreur et la vérification d'erreur de données sont accomplies au début du procédé d'accès au fichier tandis que la formation du 20 code de vérification d'erreur et l'enregistrement sont accomplis à la fin du procédé d'accès au fichier. Cependant, les premières étapes peuvent être accomplies lorsqu'une commande de lecture est fournie par le terminal tandis que les dernières sont accomplies 25 lorsqu'une commande d'écriture est fournie par le terminal. Dans ce cas, l'organigramme est formé par suppression des étapes 20 et 26, de celui montré à la figure 2.

Si la correction de données est également 30 accomplie après que la vérification des erreurs de données d'enregistrement a été accomplie, le syndrome qui est le résultat de la vérification ci-dessus peut être décodé, permettant ainsi de corriger les erreurs des données. Dans ce cas, le procédé retourne, après le 35 traitement des erreurs à l'étape 28, à l'étape 24.

Il faut noter qu'en accomplissant le code de vérification d'erreur sur la carte à circuits intégrés, il en résulte deux bénéfices significatifs. Avant tout, le terminal n'a pas à être chargé par l'accomplissement 5 de la vérification d'erreur qui nécessiterait la lecture de la totalité des données dans le bloc d'applications auquel il a accès avant qu'un calcul de vérification d'erreur ne puisse être accompli. Au contraire, la fonction de vérification d'erreur est accomplie 10 uniquement dans la carte à circuits intégrés, sans charger le terminal. Il est également significatif de noter que le code de vérification d'erreur est accompli sans prévoir un accès externe au code de vérification d'erreur lui-même et ainsi en inhibant l'opportunité que 15 des personnes non autorisées n'altèrent le code de vérification d'erreur et ainsi n'altèrent les données. Non seulement la vérification d'erreur est accomplie sur la carte à circuits intégrés elle-même, et non seulement elle est accomplie de manière que le code de vérification 20 d'erreur soit inaccessible à des dispositifs externes, mais elle est par ailleurs accomplie dans une carte à circuits intégrés d'utilisation générale, capable d'accéder à diverses applications et où les applications peuvent demander des blocs de mémoire d'applications de 25 tailles variables, chacun ayant son propre code protégé de vérification d'erreur.

R E V E N D I C A T I O N S

1. Carte à circuits intégrés adaptée à une interface avec au moins un terminal pour accomplir un certain nombre d'applications, caractérisée en ce qu'elle comprend en combinaison :

5 un moyen formant mémoire (3, 4) divisé en un certain nombre de blocs d'applications pour le stockage de l'information se rapportant aux applications, certains des blocs d'applications étant d'une taille différente
10 des autres, le moyen formant mémoire ayant également un bloc protégé pour le stockage d'une information de bloc d'application comprenant des données de répertoire et un code de vérification d'erreur se rapportant à chacun des blocs d'applications comprenant des données de répertoire
15 et un code de vérification d'erreur se rapportant à chacun des blocs d'applications ;

20 un microprocesseur (2) pour contrôler l'accès à la mémoire de manière à empêcher l'accès au bloc protégé par le terminal (5) et à sélectivement permettre l'accès aux blocs d'applications par le terminal pour accomplir sélectivement les applications ; et

25 un moyen formant programme pour forcer le microprocesseur

30 (a) à calculer un code de vérification d'erreur pour un groupe de données se rapportant à l'une parmi un certain nombre d'applications et à stocker le groupe de données et le code calculé de vérification d'erreur dans un bloc choisi d'applications et le bloc protégé dans l'emplacement associé au bloc choisi d'applications respectivement lorsque le groupe de données est envoyé par le terminal,

(b) à détecter l'erreur de donnée dans le groupe stocké de données dans un bloc d'applications assigné du terminal en utilisant le groupe de données et le code de vérification d'erreur lorsque le bloc de données est lu au terminal.

5 2. Carte selon la revendication 1, caractérisée en ce que le bloc protégé (31) comprend un certain nombre d'emplacements de mémoire, un pour chaque bloc d'applications, le bloc protégé étant configuré de telle 10 manière que l'information en rapport aux blocs d'applications associe les données de répertoire au code de vérification d'erreur pour chaque bloc respectif d'applications.

15 3. Carte selon la revendication 2, caractérisée en ce qu'elle comprend de plus un moyen répondant à une commande ouverte pour avoir accès à un fichier d'application particulière pour initialiser l'étape de détection (b) et un moyen répondant à une commande fermée pour terminer l'accès à un bloc d'applications pour 20 appeler l'étape de calcul (a).

25 4. Carte selon la revendication 3, caractérisée en ce qu'un moyen est prévu pour détecter si les données dans un bloc d'applications ont été écrites et en réponse à cela appeler l'étape (a) de calcul de vérification d'une erreur sur les données réécrites.

30 5. Carte selon la revendication 2, caractérisée en ce qu'elle comprend un moyen répondant à la réception d'une commande de lecture du terminal pour appeler l'étape (b) de détection d'une erreur de données et un moyen pour répondre à une commande d'écriture du terminal pour appeler l'étape (a) de calcul de vérification d'erreur.

6. Procédé d'actionnement d'une carte à circuits intégrés à applications multiples, la carte ayant un microprocesseur et une mémoire, caractérisé en ce qu'il consiste à :

5 (a) subdiviser la mémoire en un bloc protégé et un certain nombre de blocs d'applications en relation avec la pluralité d'applications, au moins certains des blocs d'applications étant de dimensions différentes les uns des autres,

10 (b) mémoriser les informations de blocs d'applications dans le bloc protégé comprenant des données d'instructions indiquant les emplacements des blocs d'applications respectifs associés à un code de vérification d'erreur relatif à la donnée stockée dans les blocs d'applications,

15 (c) répondre à une requête d'accès à un bloc d'applications choisi en localisant le bloc utilisant la donnée de répertoire stockée et accomplir une vérification d'erreur sur la donnée stockée en utilisant 20 le code de vérification d'erreur correspondant au bloc d'applications choisi, et

25 (d) répondre à un fonctionnement dans lequel la donnée dans un bloc d'application choisi est modifiée en accomplissant un calcul de code de vérification d'erreur sur la donnée dans le bloc choisi et stocker le code de vérification d'erreur dans la zone protégée dans l'emplacement associé au bloc choisi.

30 7. Procédé selon la revendication 6, caractérisé en ce qu'il consiste de plus à tester la performance d'une opération d'écriture et à accomplir l'étape (a) lorsqu'une opération d'écriture est détectée.

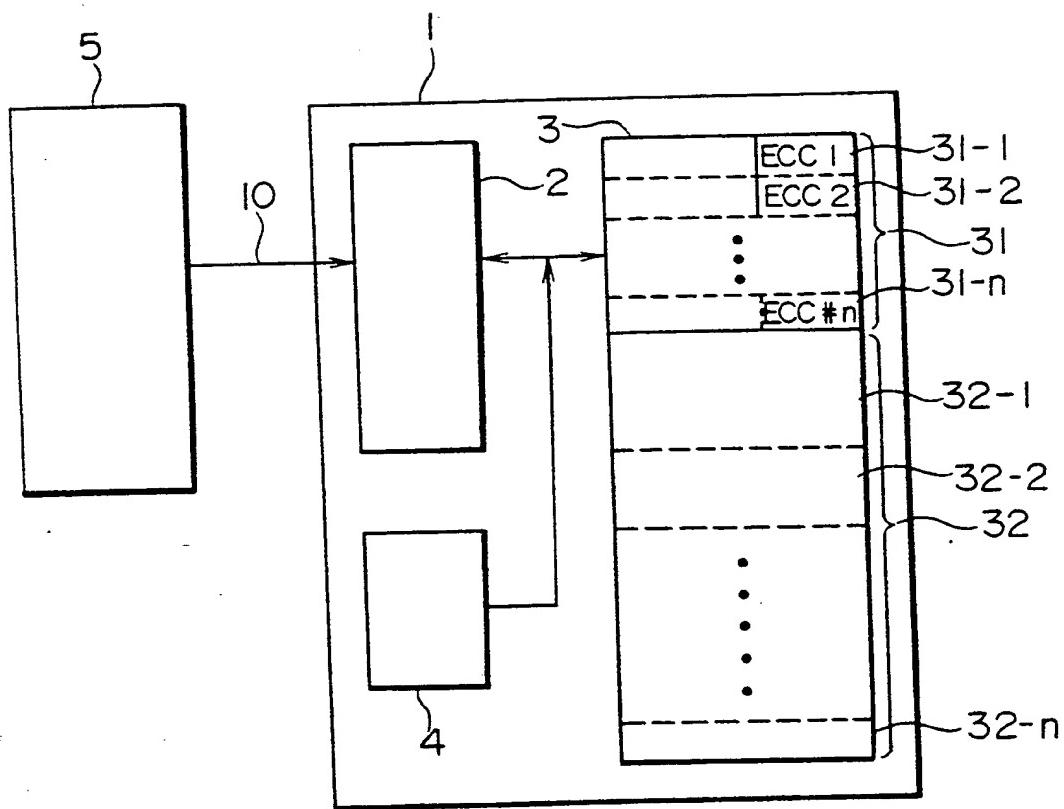
35 8. Procédé selon la revendication 6, caractérisé en ce que l'étape (b) est appelée après que toutes les opérations de lecture et d'écriture soient terminées dans une opération d'accès à un fichier.

9. Procédé selon la revendication 6,
caractérisé en ce que l'étape (c) précitée est accomplie
en réponse à une commande de lecture dirigée au bloc
d'applications choisi et l'étape (d) est accomplie en
liaison avec une commande d'écriture dirigée au bloc
d'applications choisi.

2612316

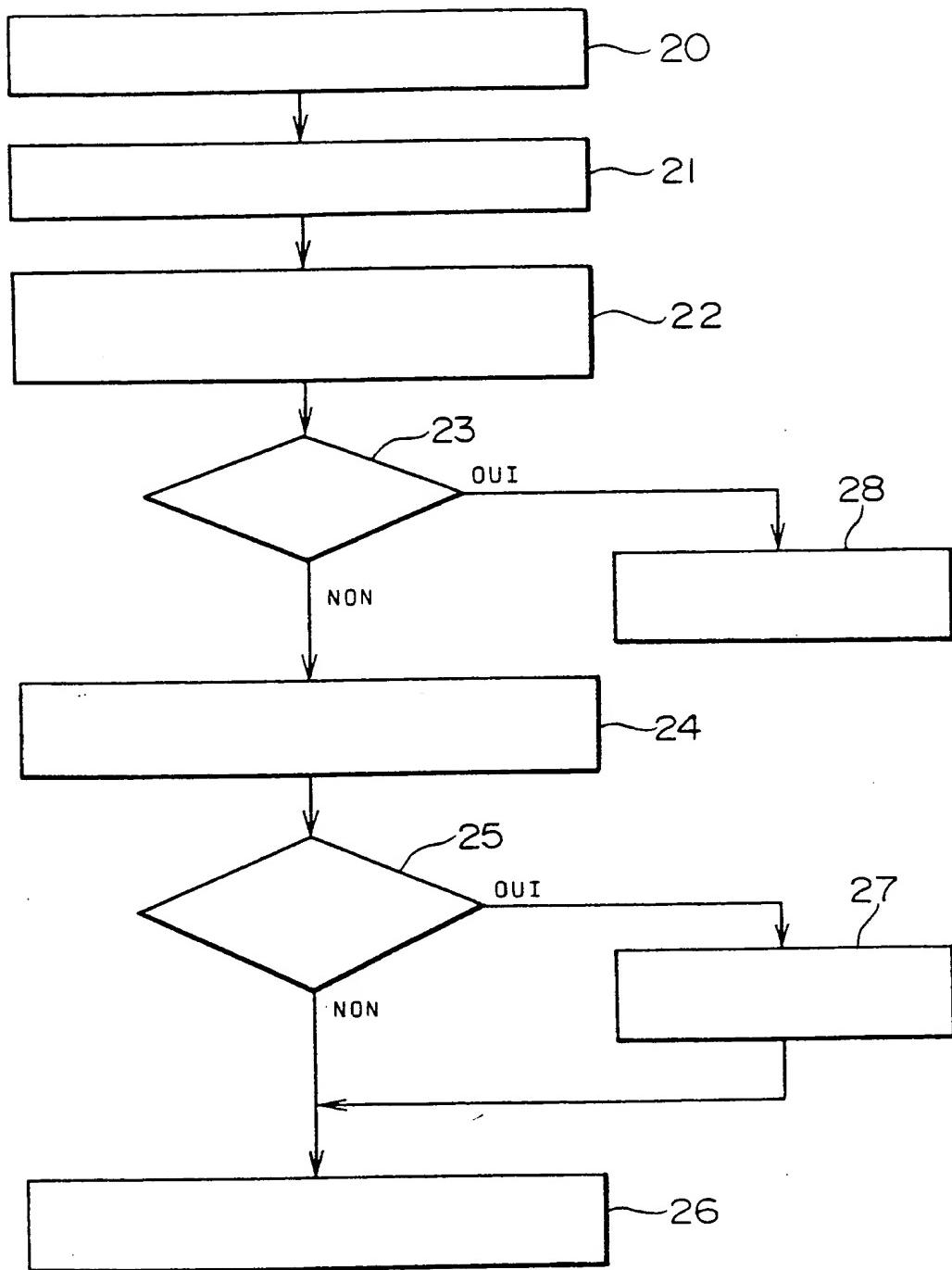
1
2

FIG.1



$\frac{2}{2}$

FIG. 2



This Page Blank (uspto)